REMARKS

An information disclosure statement summarizing most of the papers written by Oscar Pastor and his colleagues is enclosed. This IDS is more complete than the IDS submitted in May 2005 and should be used in conjunction with the CD-ROM submitted with the IDS submitted in May 2005 which contains copies of the papers and office actions cited. A new 3-page 1449 form with correct serial numbers and filing dates on page 2 has been supplied.

At the outset, claims 4 and 10 have been voluntarily amended to make it clear that the claim is talking about automatic generation of computer code from the Formal Language Specification.

Some of the dependent claims have been canceled voluntarily to make room for some new claims directed to the translation process to restate this process in different terms and claim aspects of it which are not disclosed in the prior art papers and which need to be claimed.

Turning to the Examiner's objections to the claims, the complained of words have been searched for and eliminated except for the word "essential". This limitation is not vague and what is essential in a conceptual model is defined in the specification in the validation section starting at page 40 and in the correctness and completeness sections starting on pages 42 and 45 in particular. Also, the term "filters" was not removed or changed as filters are clearly supported and defined in the specification at several locations: p. 19, line 17 where it is taught that every class has filters as part of it (hereafter page and line numbers are noted as for example 17/9); 20/22 filters are formulas; etc. A seach of the specification for filters will find many teachings about what filters are and where they are used and why. In claim 25, the term filter is used twice. This is a validation claim. The validation section of the specification describes filters as used for static and dynamic integrity constraints. The Presentation Model also uses filters as part of the user interface patterns. See for example in the Presentation Model portion of the specification starting at page 29, filters are described in the Population Selection Pattern user interface mechanism at 30/11 and in other user interface mechanisms. Filter Expressions as formulas are taught at 31/25.

The terms used in these claims are described in the specification, so their meanings can be easily derived by study of the specification.

The term "formal language specification" has been amended in all claims so as to be capitalized.

The term "he or she" has been changed to "said user" in each claim in which this terminology appears.

In response to the Section 112 rejection, claim 1 has been amended to specify a mechanism to determine a user's privileges from log in information supplied by the user that identifies the user. Also, "necessary arguments" has been changed to "arguments required" for the service to execute.

The term "take action" used with respect to checking integrity constraints has been replaced with "reversing any changes in state which caused said integrity constraints to be not satisfied". This all-or-nothing policy is supported in the specification at 39/2.

The undersigned did not find any other occurrences of "this user's privilege level" and believes that "a user's privilege level for a user identified by user name and password" (Claim 2) is not indefinite. Those skilled in the art who have read the specification understand that when a user logs in and gives a user name and password, that user's identity and privilege level in terms of what objects he can see and query and what services he can invoke is conventional and not indefinite. There is nothing indefinite about "a user".

The phrase "carry out appropriate action" with respect to trigger events having occurred has been changed in every claim where it occurred to "invoke a predetermined service associated with a trigger event " to make it more clear what happens with a trigger event occurs.

The term "perform step B" has now been supplied with antecedent basis by changing b) to B) in the designation of steps for parent claim 26.

New claims 40 - 63 have been added to claim the validation and translation process to convert the Conceptual Model, as encoded in a Formal Language Specification into full, complete working code which implements a computer program with functionality as defined in the Conceptual Model and a User Interface as defined in the Presentation Model part of the Conceptual Model.

The Section 101 Rejection

Claims 3 and 33 have been amended to "physical computer-readable storage media". Claim 7 has been cancelled for other reasons not related to this rejeciton

The Prior Art Rejections

The claims have been rejected as anticipated over US 5,481,718.

Claims 1, 10, 13, 14, 18-25 and 30-32:

The limitations in claim 1 and also in claims 10, 13, 14, 18 to 25 and 30 to 32 that the examiner finds anticipated by Ryu focus on how a Formal Specification Language which encoding a Conceptual Model that represents the requirements of a full and complete program or software system is:

- validated so as to ensure correctness and completeness, and
- automatically transformed into source code corresponding to said software system which will be functionally equivalent to the Formal Language in that it performs the following functions:
    - o User authentication and validation
    - o Determination of user privileges
    - o Match of Service Requests Against Objects Implementing Services
    - o Collection of Service Arguments
    - o Check for the Validity of State Transitions
    - o Verification of Preconditions
    - o Valuations Calculation
    - o Check of Integrity Constraints
    - o Check of Trigger Relationships

The examiner cites passages in Ryu that allegedly anticipate said limitations but they do not actually anticipate them as it will be shown in the following sections:

<u>Validating a Formal Language Specification</u>

According to the examiner, col. 10, lines 42-51 and also FIG. 10 and the related discussion anticipates: "*validating a formal language specification written in a formal language which has predetermined rules of syntax and semantics, said formal language specification defining a computer program to be automatically written*"

Ryu, Col. 10, lines 42-51 teaches that: "A simulation is carried out with respect to the generated object so as to determine whether or not a correct function is actually obtained, or a provisional operation is carried out with respect to the generated objects for which the simulation is ended. Such a process corresponds to the provisional operation mode 216 shown in FIG. 10. The dynamic object processing unit 212 uses the content of the component attribute file 205 to simulate the corresponding processing operation."

The prosecuted application teaches the use of formal language specifications "having rules of syntax and semantics" and then validation of said formal language specifications according to said rules.

Ryu, however, does not teach the use of a formal language specification nor any other type of formalism. No evidence can be found in Ryu that mentions syntax or semantics of any specification as being part of his invention.

Furthermore, notice that "validation" and "simulation" are not synonyms nor they are interchangeable terms.
"**Validation**" is "the process of checking if something satisfies a certain criterion" (source: http://en.wikipedia.org/wiki/Validation) while "**simulation**" is "an imitation of some real device or state of affairs. Simulation attempts to represent certain features of the behavior of a physical or abstract system by the behavior of another system." (source: http://en.wikipedia.org/wiki/Simulation). No evidence on "validation" can be found in Ryu.

Therefore, since validation and simulation are not synonyms, and Ryu does not teach the use of formal language specifications, Ryu does not anticipate the validation of formal language specifications disclosed in the prosecuted application.

Furthermore, the test process taught by Ryu in Col. 10 lines 62-col. 11 line 2 ("The dynamic object processing unit 212 uses the content of the component attribute file 205 and temporarily carries out a substitute process with respect to a predetermined process. On the other hand, if becomes necessary to carry out a test process, the dynamic object processing unit 212 activates the direct object expansion unit 215 to generate the executionable process data 214 and carries out this process. In FIG. 10, such a process mode is indicated as the instant operation mode 217.") relates the testing activity to "executionable process data" and not to "formal language specifications". So even though both the testing in Ryu and the validation in the prosecuted application disclose the automatic feature, such feature does not apply to the same subject matter.

<u>Translating a Formal Language Specification into a Full, Complete Computer Program</u>

According to the examiner, the rest of references to passages in Ryu show anticipation of the limitation of automatically translating a Formal Language Specification into a full and complete computer program that "needs no additional third party source code or source code from existing components or code libraries to be compiled with it to make said computer program complete and which implements the requirements of said Formal Language Specification".

Specifically, the examiner cites the following passage as anticipating this limitation in Col. 7 to 8, lines 62-67 to 1-5: "FIG. 7 shows a static world 410, a dynamic world 420 and a functional model 431. Thus, the system describes the real world in terms of the object model wherein the real world is described by the external definition and the internal definition such that (i) the internal definition forms a concealed area acting as a functional model 431 that stores therein existing methods 313 or classes 302. The class 302 may include newly created methods and classes. Further, the external definition is incorporated into the system in the form of the static world 410 and the dynamic world 420."

The Formal Language Specification in the prosecuted application comprises an Object Model, a Functional Model a Dynamic Model and a Presentation Model. **The presentation model is not found and therefore not anticipated by Ryu, and is one of the limitations that have to be present in order to have a Formal Language Specification translated into a full and complete computer program, because said Presentation Model, as disclosed in the prosecuted application is the representation of the desired user interface of said computer program.** For this reason alone, claim 1 and all claims that include the presentation model are not anticipated by Ryu. A computer program without user interface would not be complete and therefore the lack of Presentation Model in Ryu's invention evidences that this limitation is not anticipated by Ryu.

Furthermore, neither the object model nor the dynamic model nor the functional model, as disclosed in the prosecuted application, are present in the Ryu system. The Examiner's prima facie case is only a case of coincidence in nomenclature whereas the actual functionality of these models is not actually present in Ryu.

Ryu's **Object Model** is "a plurality of objects combined with each other, each of said objects being formed of data and methods" (see Abstract: "An object-oriented data processing system performs a desired processing based upon an object model including a plurality of objects each formed of data and methods" and also col. 5 lines 25-30) The prosecuted application teaches an object model comprising classes and relationships (aggregation, inheritance, agent) between them, said classes comprising attributes (constant, variable or derived), an identification function, services (events or transactions), derivations and integrity constraints. Even though the attributes and services of classes in the prosecuted application could be seen as the "data and methods" of the objects in Ryu, the rest of primitives comprised in a class (derivations, integrity constraints, ...) are not found in Ryu and therefore not anticipated.

Ryu's **Dynamic Model** is the instantiation of what is expressed in the static model. See Abstract ("a dynamic model that indicates the time sequential relationship between instances forming the classes as a session"), Col. 8 lines 8-11 ("Further, the dynamic world 420 includes the instances corresponding to the classes such that the instances are serially connected according to the order of processing, to form a session.") and Col. 9 lines 49-53 ("The dynamic model is formed by creating an instance by allocating instance data to the foregoing class 302 and designing the time sequential relationship between the instances thus created. Further, the restriction of causality is imposed between the static model and the dynamic model."). The static model describes a set of classes and relationships between them, whose instantiation in terms of instances of classes forms the dynamic model. That is, the dynamic model is the "run-time" version of the static model, and the terms "static" and "dynamic" are used in this sense in Ryu and is depicted in Fig 9.
The prosecuted application, on the other hand, teaches that the Dynamic Model comprises a State Transition Diagram for each class in the Object Model, and an Object Interaction Diagram where Triggers and Global Transactions are defined. The Dynamic Model in the prosecuted application is then radically different, other that in its name, from the Dynamic Model in Ryu. The concepts of State Transition Diagram, State Transition, Trigger, and Global Transaction are not taught, hence not anticipated, by Ryu and therefore although both the prosecuted application and Ryu teach the use of a Dynamic Model they clearly refer to different things. No anticipation exists.

Ryu's **Functional Model** is a set of classes and methods related to existing behavior. See Col. 16 lines 25-32 ("As mentioned above, existing and/or newly created methods are combined by the operation of the class schema and the instance schema, and a system is constructed as desired according to the processing demand. In doing so, it is necessary to arrange such that any person who wishes to use the methods and

classes forming a functional model in the concealed world 430, can know about the function of these methods and classes at a later time.")
The prosecuted application, on the other hand, teaches that the Functional Model comprises a set of valuations, defining how the occurrence of an event of a class affects the values of variable attributes of said class. The Functional Model in the prosecuted application is then radically different, other that in its name, from the Functional Model in Ryu. **The concept of Valuation is not taught, hence not anticipated, by Ryu and therefore although both the prosecuted application and Ryu teach the use of a Functional Model they clearly refer to different things.**

Furthermore, even though Ryu teaches the use of class attributes and event [objects], it does not teach how the occurrence of the latter affect the values of the former. Furthermore, the term "event" in Ryu is always coupled with the word "object" so they are referred to as "event objects" which are a special type of objects (see Col 3, lines 20-25: "Therefore, the composite objects are formed by successively combining the objects. For example, the so-called primitive objects which will be described later are combined to form a capsule object, the capsule objects are combined to form an event object, and the event objects are combined to form a system object.") while in the prosecuted application events are atomic services of classes (in other words, services whose execution is atomic) but not a special kind of object.

Also, the examiner cites passages from Ryu as anticipating the steps comprised in the above limitation in the prosecuted application.

<u>Write Code to Authenticate Users</u>
<u>Write Code to Determine User Privileges</u>
<u>Write Code to Match Service Requests to Objects Implementing Services</u>
<u>Write Code to Collect Service Arguments</u>

These four limitations are said to be anticipated by the passage in col. 38 to 39, lines 55-67 to 1- 31:

"FIG. 46 shows the object-oriented system designing that forms the background of the present invention, wherein FIG. 46 shows the static world 410, dynamic world 420 and the functional world 431 similarly to FIGS. 8 and 9.

Referring to FIG. 46, the process proceeds as follows.

(1) The user creates a request about the system design.

(2) Description is made about the problems related to the request, and a scenario for defining the specification is created.

(3) Next, an analysis is made about the data and methods that are needed as well as the data and methods that are available.

(4) Based upon the analysis, the extraction of necessary classes is made in the static world 410, and the entity-relation (E-R) charts such as the one shown in FIG. 2 are created for the necessary data and necessary methods. Further, the methods for the schema are selected based upon the E-R chart thus obtained. Furthermore, the data and the methods are properly related based upon the E-R chart of the methods.

(5) Once the E-R chart is obtained, the classes are created and registered as indicated in FIG. 7, 12, 13 or 14. Further, composite classes are created.

(6) In the dynamic world 420, a flowchart is created about the related events, and the events that can be processed parallel are determined. Further, the instances are created as indicated in FIG. 7, 12, 13 or 14, and the sequence is designed. Further, the causality is extracted in relation to the determination of the events that can be processed in parallel, and the causality is represented in the static world in the form of the state table 414 as indicated in FIG. 7 or FIG. 28.

(7) Based upon the instance thus obtained, it becomes possible to meet the request of the user. Further, the classes and the composite classes thus obtained are kept as the functional model 431 for future reuse. In addition, those methods that are needed for the creation of the class but not existing, are created, and the methods thus

created are also registered for future reuse. The creation of the missing methods is achieved by the processing shown in FIG. 7, 10 or ! 1 that corresponds to the processing of the hyper language processing unit 222, wherein an improvement is made based upon the provisional operation conducted for various combinations of the components."

This passage teaches how a user of Ryu's invention interacts with it in order to create a design of an object-oriented system. So the purpose of the user is to create a design of an object-oriented system or enhance an existing design (see step (1) "The user creates **a request about the system design**.") Whereas in the prosecuted application the limitations said to be anticipated by this passage in Ryu, the user is that of a software system or computer program which was created in an automated way from a Formal Language Specification. Therefore, the user in these limitations of the prosecuted application is interacting NOT with the design of a system, but with the system itself.

Furthermore, the outcome of the process referred to in FIG. 46 in Ryu is the creation of classes, composite classes and methods (see step (7) "the classes and the composite classes thus obtained are kept as the functional model 431 for future reuse. In addition, those methods that are needed for the creation of the class but not existing, are created, and the methods thus created are also registered for future reuse.") Whereas in the prosecuted application, the outcome of the interaction of the user with the system is NOT the creation of classes or methods but the invocation of (existing) services of (existing) classes to change the state of a set of objects of the system.

More specifically these four limitations refer to the writing of computer code that, being part of the computer program or software system automatically obtained from a Formal Language Specification, provides the following functionality:

- Authentication of users of the computer program or software system. This being a security mechanism in order to prevent unauthorized users of the system from using it.
- Determination of privileges of any given authorized user. Said privileges consisting of the set of attributes of any class in the system said user will be allowed to query and the set of services of the system said user will be allowed to invoke.
- Matching of service requests to objects implementing said services. This being a sort of mediator between the user or process invoking a service and the actual object implementing the functionality of the invoked service.
- Collection of service arguments. This being part of the invocation of any service which will require values for each of its arguments to be provided by the user or process invoking said service.

In summary, these four limitations refer to the writing of code of a software system automatically obtained from a Formal Language Specification that provide the functionality related to user interaction with the system in terms of authentication, determination of privileges and invocation of services by providing values to all arguments of the invoked services. Therefore, since Ryu discloses that users interact with designs of object-oriented systems and not with computer programs or software systems automatically created from a Formal Language Specification, these four limitations are not anticipated by Ryu.

<u>Write Code that Displays Means to Invoke Services</u>
<u>Write Code to Check the Validity of State Transitions</u>

These two limitations are said to be anticipated by the passage in col. 31, lines 47-67: "Referring to FIG. 37, a call is set up in response to the request of call from the calling party, and a connection is established upon the response from the called party. After communication, the connection is disconnected and the channel released. Further, the call number is released.

In the present invention, it becomes possible to transmit data at high speed between the terminals, by way of the foregoing three layered communication processing system, whenever and each time there occurs a need in a terminal to request missing data. In such a system, there may be a case wherein the terminal that issues the request already have information about the terminal that manages the requested data. In such a case, the broadcasting of inquiry via the control line 152 is not necessary and may be omitted.

As set forth above, the present invention provides a distributed data processing system wherein a number of terminals each having a program and/or database are connected with each other by a LAN or other communication network, such that a terminal therein carries out a processing based upon transfer of program and/or data from another terminal whenever such a need occurs."

However, if we look at col. 31, lines 33-35 ("FIG. 37 shows the control procedure for connecting a phone in the ISDN, wherein the definition of the symbols therein is as follows.") it is evident that what Ryu teaches in this passage, in FIG. 37 and in its related discussion does not anticipate any of these two limitations in the prosecuted application. Ryu discloses "the control procedure for connecting a phone in the ISDN" being part of a "Broadcast System" (see col. 28, lines 24-30), which is one of the data processing systems that execute a processing in a terminal which Ryu cites as examples of embodiments of his invention (see col. 27, lines 32-36: "There are several known systems that execute a processing in a terminal by receiving necessary information from another terminal or center station. Further, there are systems that ask another terminal to execute a process by sending thereto the information necessary for processing.")

Hence, embodiments of Ryu's invention are data-processing systems whereas any embodiment of the prosecuted application is a Software Production System that allows the creation of a Formal Specification Language which is automatically validated and then automatically transformed into a computer program. One part of said computer program being code that displays "means by which an entity can invoke a service, and which receives input to invoke a particular service and responds by sending a message to the appropriate object server to invoke the service, said message including the necessary arguments for the service to execute". What Ryu discloses is a mechanism to communicate terminals (namely a mechanism based on the use of fiber optics, LANs and the like) The prosecuted application does not claim anything related with the mechanisms used to communicate entities invoking services with objects actually implementing said services, but rather what messages (and which contents they have) are sent back and forth (regardless of the communication mechanisms used to transmit said messages).

The second limitation which is said to be anticipated by Ryu in the passage found col. 31, lines 47-67 refers to the writing of computer code that, being part of the computer program or software system automatically obtained from a Formal Language Specification, provides the functionality to check the validity of state transitions in objects the state of which is to be acted upon by the requested service. State transitions are part of the State Transition Diagram. As disclosed in the prosecuted application: "Classes also have a State Transition Diagram which is a set of states and transitions between them. Each state transition is related to an action (service plus list of possible agents) that can change the state of the object." And furthermore: "The state transition diagram (STD) is used to describe correct behavior by establishing valid object life cycles for every class. A valid life refers to an appropriate sequence of states that characterizes the correct behavior of the objects that belong to a specific class. Transitions represent valid changes of state."

Neither the concept of state transition nor the concept of State Transition Diagram is taught by Ryu and therefore not anticipated. What Ryu teaches in col. 31, lines 47-67, as set forth is related with a mechanism to communicate terminals, not with the writing of code that checks the validity of state transitions.

Write Code to Verify Preconditions
Write Code to Perform Valuation Calculations

These two limitations are said to be anticipated by the passage in col. 19, lines 55-67:"FIG. 21B shows the flowchart related to the process of FIG. 21A. The steps of FIG. 21B is as follows.
(S8) Check the "part-of" hierarchical code in the object command and conduct a search in the semantic conversion table 360 to obtain a corresponding class name.
(S9) Conduct a search of the second correspondence table mentioned above based upon the obtained class name to obtain the relationship between the parent class and the children classes.
(S10) Construct a hierarchical class from the parent class and the children classes thus obtained.
FIGS. 22A and 22B are diagrams for explaining the relationships among a plurality of classes."
This passage in Ryu teaches how to deal with "part-of" relationships, which define hierarchies of classes (a class can have descendent classes, said descendent classes being related with their parent class with a "part-of" relationship, which can in turn have other descendents). Ryu teaches that the "object command" allows to obtain a "class name" by conducting a search "in the semantic conversion table 360". Said "semantic conversion table" is illustrated in FIG. 21 A and shows a one-to-one mapping between object commands and class names so a search in the semantic conversion table of a given object command will provide the corresponding class name (which is what Ryu teaches in the step (S8) in the passage above). By

examining all the "part-of" relationships of said obtained class name, the object commands of children classes are obtained and by conducting searches of the semantic conversion table, the class names of children classes are thus obtained. Finally, once the names of both the parent and children classes have been obtained, a new class referred to as "hierarchical class" is created "from the parent class and the children classes". Therefore, what this passage of Ryu discloses is how to construct a "hierarchical class" from the object commands and "part-of" relationships between a parent class and its children classes, this being directly related with the notion of inheritance.

The two limitations above of the prosecuted application, however, do not teach anything about the notion of inheritance:
- The first of the two limitations refers to the writing of computer code that, being part of the computer program or software system automatically obtained from a Formal Language Specification, provides the functionality to check that preconditions of a service hold. A precondition being a well-formed formula stating a Boolean condition that must be true in order for a service to be executed. ("Preconditions are formulas that need to be satisfied in order to execute the corresponding action." and "conditions that must be satisfied in order to allow the execution of a service, which are called "preconditions".")
- The second of the two limitations refers to the writing of computer code that, being part of the computer program or software system automatically obtained from a Formal Language Specification, provides the functionality to perform valuations calculations. A valuation is a primitive in the Formal Specification Language that defines how the occurrence of an event (one of the two types of services disclosed in the prosecuted application) affects the value of variable attributes of the class owning said event. ("Each attribute can also include a list of valuations, which are formulas that declare how the object's state is changed by means of events. Valuation formulas are structured in the following parts: condition (that must be satisfied to apply the

effect), event and effect of the event to the particular attribute.")

Therefore, these two limitations are not anticipated by the cited passage of Ryu, which is clearly related with the notion of inheritance and not with the notions of precondition and valuation. Furthermore, said notions of precondition and valuation, as disclosed in the prosecuted application are not taught by Ryu and therefore none of these two limitations is anticipated by Ryu.

Write Code to Check Integrity Constraints
Write Code to Check Trigger Relationships

These two limitations are said to be anticipated by the passage in col. 9-10, which is not reproduced here for the sake of brevity.

The passage describes the "static world" and the "dynamic world" represented in Ryu's invention by means of the "static model" and the "dynamic model" which, as explained above, do not anticipate the concepts of "Object Model" nor "Dynamic Model" as disclosed in the prosecuted application. The passage also describes "the relationship between the static model, dynamic model and the functional model", said "functional model" in Ryu not anticipating, as discussed above, the "Functional Model" of the prosecuted application.

Furthermore, the two limitations above of the prosecuted application do not teach anything about the "static world" and the "dynamic world" nor about how the "static model, dynamic model and functional model" relate to each other. Rather:
- The first of the two limitations refers to the writing of computer code that, being part of the computer program or software system automatically obtained from a Formal Language Specification, provides the functionality to check integrity constraints. The prosecuted application discloses integrity constraints as being well formed formulas

that define a condition on the values of attributes of objects of a class, said condition having to hold for every object of a class to guarantee the integrity (hence, the name "integrity constraints") of the state of objects.

- The second of the two limitations refers to the writing of computer code that, being part of the computer program or software system automatically obtained from a Formal Language Specification, provides the functionality to check trigger relationships. The prosecuted application teaches that triggers are a type of object interaction. Specifically triggers are defined as "object services that are automatically activated when a pre-specified condition is satisfied".

Ryu does not teach the notion of integrity constraints. The only mentions to "integrity" being those that refer to "data integrity". Ryu acknowledges that maintaining data integrity in a data-processing system is a problem, but provides a solution to this problem (col. 28, lines 49-52: "the foregoing approach to carry out data management in each of the terminals such that the terminal manages the data that belongs thereto, provides the solution to the problem of maintaining proper data integrity.") which does not anticipate the definition of well-formed formulas that have to be checked by the code which is written according to the first of the two limitations above. Therefore the first of the two limitations above is not anticipated by Ryu.

Ryu does not teach either the notion of trigger relationships and therefore the second of the two limitations above is not anticipated by Ryu either.

Claims 2 and 4:

Since claims 2 and 4 are the apparatus claim version of process claim 1, these claims are not anticipated for the same reasons detailed above with respect to claim 1.

Claim 3:

Since claim 3 is the computer product claim corresponding to claim 1, it is not anticipated for the same reasons claim 1 is not anticipated.

Claims 5-8, 11-12:

These claims have been cancelled for reasons not related to the prior art rejection.

Dependent Claims 15 - 25:

These claims all depend from claim 14 which is a process for validating a Formal Language Specification, standing alone. The arguments why claim 14 is not anticipated by Ryu were given in great detail above under the heading **Claims 1, 10, 13, 14, 18-25 and 30-32**. Dependent claims 15-25 are not anticipated for the same reasons their parent claim 14 is not anticipated.

Claims 26 - 32:

This claim is an apparatus claim covering a computer programmed to perform validation on a Formal Language Specification in the same way that process claim 14 performs validation. Accordingly, claim 26 is not anticipated for the same reasons claim 14 is not anticipated and as explained in great detail above. Claims 27 - 32 are dependent claims which depend from claim 26 and are not anticipated for the same reasons claim 26 is not anticipated.

Claims 33 - 39

Claim 33 is an independent computer readable medium claim wherein the medium stores computer readable instructions to perform the validation process of claims 14, 15, 16, 24, 20 and 24 respectively. Therefore, these claims are not anticipated for the same reasons claims 14, 15, 16, 24, 20 and 24 respectively are not anticipated.

Respectfully submitted,

Dated: December 15, 2005

[illegible]

Ronald Craig Fish
Reg. No. 28,843
Tel 408 778 3624
FAX 408 776 0426

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail, postage prepaid, in an envelope addressed to: Mail Stop ~~RCE~~ PATENT APP, Commissioner for Patents , P.O. Box 1450, Alexandria, Va. 22313-1450.

on 12/15/05
(Date of Deposit)

[illegible]

Ronald Craig Fish, President
Ronald Craig Fish, a Law Corporation

Reg. No. 28,843